第十八节 高可用架构升级

Spring Cache 抽象

基本实现模式

缓存操作注解

@Cacheable

@CacheEvict

@CachePut

缓存操作元数据 API - CacheOperation

@Cacheable - CacheableOperation

@CacheEvict - CacheEvictOperation

@CachePut - CachePutOperation

缓存操作元数据来源 - CacheOperationSource

注解实现 - AnnotationCacheOperationSource

缓存注解解析器 - CacheAnnotationParser

具体实现 - SpringCacheAnnotationParser

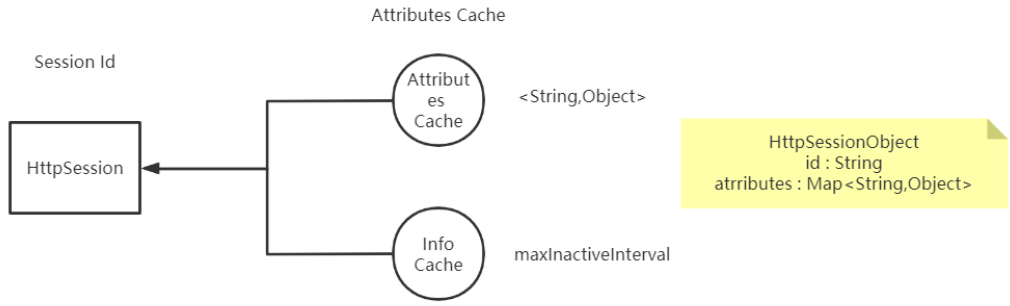
```
@Service
public class BookService { // 被代理的 Bean

    @Cacheable(cacheNames="books", key="#isbn")
    // "books" Cache, Key = "#isbn" -> JCache - Cache<ISBN, Book>
    //                              -> Spring Cache - Cache
    public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed) {

    }

    public Book getBook(...){ // CacheOperation 集合 == 空

    }
}
```

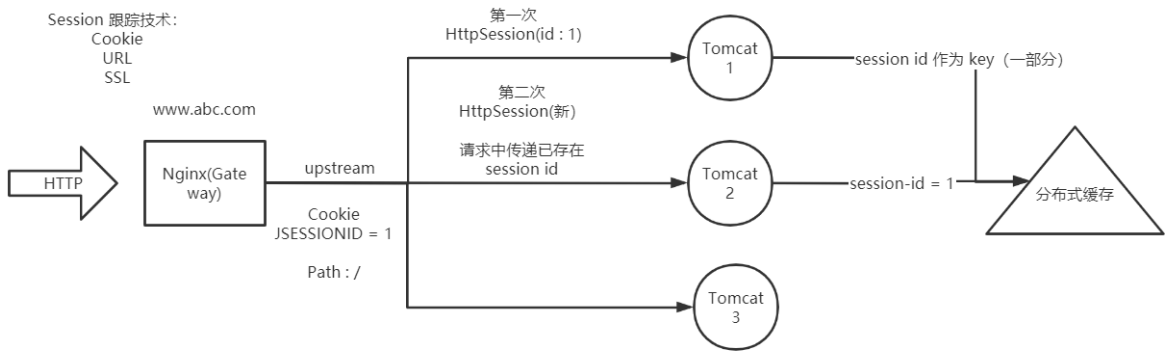
Spring Session

关联内容

Tomcat Session 复制

JSESSIONID = 266EEBAEAE9E64F9DC3CFA88F430EF81.abcde





相关内容

Spring 编程风格趋势（主观判断）

传统面向对象编程

OOP + GoF23 + Other Patterns

Reactive 编程风格

Reactive Streams JVM

Function Programming

Fluent API 编程风格

HttpSecurity

WebSecurity

作业解析

如何解决多个 WebSecurityConfigurerAdapter Bean 配置相互冲突的问题？

提示：假设有两个 WebSecurityConfigurerAdapter Bean 定义，并且标注了不同的 @Order，其中一个关闭 CSRF，一个开启 CSRF，那么最终结果如何确定?

背景：Spring Boot 场景下，自动装配以及自定义 Starter 方式非常流行，部分开发人员掌握了 Spring Security 配置方法，并且自定义了自己的实现，解决了 Order 的问题，然而会出现不确定配置因素。

解析：每个 WebSecurityConfigurerAdapter Bean 关联着一个 HttpSecurity 对象，而 HttpSecurity 对象是由 WebSecurityConfigurerAdapter 内部创建：

```
    protected final HttpSecurity getHttp()
throws Exception {
        if (this.http != null) {
            return this.http;
        }
        AuthenticationEventPublisher
eventPublisher =
getAuthenticationEventPublisher();
        ......
        configure(this.http);
        return this.http;
    }
```

当 WebSecurityConfigurerAdapter 子类扩展 configure(HttpSecurity) 时，HttpSecurity 不会相互影响

如果 Spring IoC 容器存在多个 WebSecurityConfigurerAdapter Bean 时，尽管可以通过 @Order 或者 Ordered 接口的方式定义优先级，优先级低会其作用。

解决方法：

1. 只允许定义一个 WebSecurityConfigurerAdapter Bean
2. 提供一个接口名字叫做 com.acme.WebSecurityConfigurer（与 Spring Security 相同）
3. 将唯一 WebSecurityConfigurerAdapter Bean 实现组合模式，组合的对象就是 com.acme.WebSecurityConfigurer 的实例，允许定义多个 com.acme.WebSecurityConfigurer Bean，分别来迭代 com.acme.WebSecurityConfigurer Bean，通过 @Order 或者 Ordered 接口来控制优先级
4. 如果出现多个 WebSecurityConfigurerAdapter Bean，通过异常或者警告来提示

> 如果 Spring Security 3.0 版本，以及对应 Spring Boot 1.x 可以使用异常来提示
>
> 如果更高的版本，只能通过警告来提示

```
public class CompositeWebSecurityConfigurer
extends WebSecurityConfigurerAdapter {

    private
List<com.acme.WebSecurityConfigurer>
webSecurityConfigurers;

    @Autowired(required = false)
    public void
setWebSecurityConfigurers(List<com.acme.WebSecu
rityConfigurer> webSecurityConfigurers) {
        this.webSecurityConfigurers =
webSecurityConfigurers;
    }

    protected void configure(HttpSecurity http)
throws Exception {
        webSecurityConfigurers.forEach( c ->
c.configure(http));
    }
}
```

作业

1. Spring Cache 与 Redis 整合

- 如何清除某个 Spring Cache 所有的 Keys 关联的对象
  - 如果 Redis 中心化方案，Redis + Sentinel
  - 如果 Redis 去中心化方案，Redis Cluster

- 如何将 RedisCacheManager 与 @Cacheable 注解打通